The zoon R package for reproducible and shareable species distribution modelling

Nick Golding[1], *Tom A. August*[2], *Tim C.D. Lucas*[3], *David J. Gavaghan*[4], *E. Emiel van Loon*[5] *& Greg McInerny*[6]

[1] *School of BioSciences, University of Melbourne, Parkville VIC, Australia*

[2] *NERC Centre for Ecology & Hydrology, Crowmarsh Gifford, Maclean Building, Wallingford, OX10 8BB, UK*

[3] *Oxford Big Data Institute, Li Ka Shing Centre for Health Information and Discovery Nuffield Department of Medicine, University of Oxford, Oxford, UK*

[4] *Department of Computer Science, University of Oxford, Oxford, UK*

[5] *Institute for Biodiversity and Ecosystem Dynamics, University of Amsterdam, Amsterdam, NL*

[6] *Centre for Interdisciplinary Methodologies, Social Sciences, University of Warwick, Coventry, CV4 7AL, UK*

Abstract

1. The rapid growth of species distribution modelling (SDM) as an ecological discipline has resulted in a large and diverse set of methods and software for constructing and evaluating SDMs. The disjointed nature of the current SDM research environment hinders evaluation of new methods, synthesis of current knowledge, and the dissemination of new methods to SDM users.

2. The zoon R package aims to overcome these problems by providing a modular framework for constructing reproducible SDM workflows. zoon modules are interoperable snippets of R code, each carrying out an SDM method, that zoon combines into a single analysis object.

3. Rather than defining these modules, zoon instead draws modules from an open, version-controlled online repository. zoon makes it easy for SDM researchers to contribute modules to this repository, enabling others to rapidly deploy new methods in their own workflows, or to compare alternative methods.

4. Each workflow object created by zoon is a re-runnable record of the data, code and results of an entire SDM analysis. This can then be easily shared, scrutinized, reproduced and extended by the whole SDM research community.

5. We explain how zoon works and demonstrate how it can be used to construct a completely reproducible SDM analyses, create and share a new module, and perform a methodological comparison study.

Introduction

Species distribution modelling (SDM) has grown rapidly over the last 20 years. It is now one of "the most widely-reviewed topics in the ecological literature" (Araújo & Peterson 2012) and the growth of this literature is still accelerating (Barbosa & Schneck 2015). The SDM software market is similarly large and diverse (Ahmed *et al.* 2015). Whilst most SDM users rely either on the MaxEnt standalone application (Phillips *et al.* 2006) or the R programming language (R Core Team 2016) as the first choice software for their analyses (Ahmed *et al.* 2015) a variety of SDM-specific R packages are available, each implementing different approaches to fitting SDMs (Thuiller *et al.* 2009; Hijmans *et al.* 2016; Naimi & Araújo 2016), as well as a range of alternative standalone applications.

As a result, the SDM community can become siloed, with groups of researchers collaborating primarily with others who use the same software, even within specific R packages (Ahmed *et al.* 2015), presenting a barrier to disseminating new SDM findings and methods. This is compounded since almost all available SDM software is focussed on analytical tasks such as constructing models, rather than enabling scientists to produce analyses in a format that others can reproduce and modify. The inability of SDM researchers to reproduce, scrutinise, and build on others' research prevents rigorous peer review, synthesis of research findings across studies, and reduces capacity for the science to be a self-correcting process (Boulton *et al.* 2012).

For example, a fundamental dispute over the ability of SDMs to detect environmental associations (Beale *et al.* 2008, 2009; Araújo *et al.* 2009) was left unresolved as only the original publication shared their analytical code (Beale *et al.* 2008). Similarly, in the past decade the SDM community has failed to carry out any large scale comparisons of SDM methods, still relying on the work of Elith *et al.* (2006) to select models and software (Joppa *et al.* 2014), despite the development of many new methods as well as improved evaluation procedures since the publication of that study (Roberts *et al.* 2016). The community's inability to repeat this comprehensive analysis is due to both lack of access to comparable data and the difficulty in learning

and applying the many different pieces of software for model fitting.

Given current software and patterns of collaboration, the SDM community is unlikely to repeat or modify analyses. Nor is it likely to produce the large model comparisons that would answer even basic questions about how best to do SDM. It may also have contributed to widespread misunderstandings about how to apply and interpret different modelling approaches (Yackulic *et al.* 2013). In order to overcome these problems, the data and code underpinning SDM research need to be made more accessible, reproducible and modifiable by the whole research community. This can be achieved if technologies enable and encourage sharing of research as fully reproducible objects (Peng 2011), in ways that suit the diversity of users involved in SDM (Ahmed *et al.* 2015).

The zoon R package has been developed specifically to improve reproducibility and comparability of SDMs in R by allowing users to encode entire SDM analyses as repeatable and extensible workflows consisting of independently executable, community-contributed modules. The module-workflow structure enables scientists to more easily create and share components of their analysis; and then access, modify, reuse and combine the components of others (see below and Figure 1). Whilst zoon's modular nature is similar to other SDM R packages such as BioMod2 (Thuiller *et al.* 2009) and sdm (Naimi & Araújo 2016), zoon instead pulls each module from an open repository that any SDM user can contribute to, and makes it easy for non-developers to contribute modules. zoon's focus on reproducible and modifiable workflows is inspired by repositories such as the Cardiac Web Lab (Cooper *et al.* 2016) and workflow systems such as Taverna (Wolstencroft *et al.* 2013) and the BIOVEL system (De Giovanni *et al.* 2015, 2016), but embeds an SDM-specific workflow system within R, making it much more accessible to the SDM community.

This paper introduces version 0.6 of the zoon R package. We describe the modular structure of zoon workflows and how they can be constructed, shared, reproduced and modified. We then illustrate how these concepts enable better SDM research by reproducing a published SDM analysis; converting a recently proposed method into a zoon module; and performing a reproducible methodological comparison.

Building a workflow

zoon encodes SDM analyses as a simple workflow of five key steps: obtaining occurrence data; obtaining covariate data; applying some processes to these data; fitting one or more models; and generating any outputs (Figure 1A). Each of these steps is encoded as one or more software modules (snippets of R code) to complete one of these tasks. zoon modules can use methods from any R package, providing a common interface between

the many SDM packages already available. Users combine these modules via a call to the `workflow` function, which executes each module in turn, before returning a `zoonWorkflow` object - a shareable, extensible and fully reproducible documentation of the SDM analysis. Figure 1B illustrates how data and outputs are passed between modules of the five different types in a a zoon workflow.

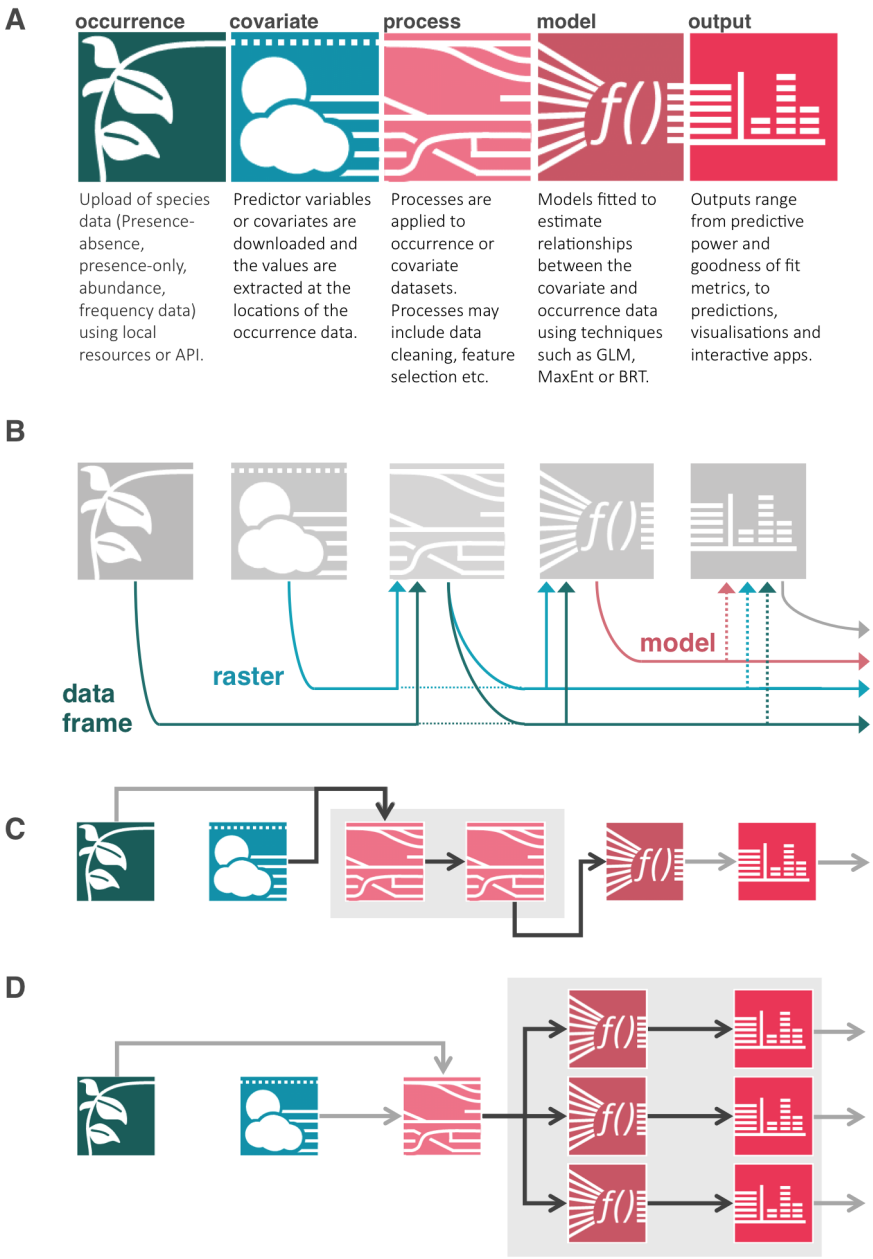

Figure 1: The modular SDM structure encoded by a zoon workflow. A) Description of the five module types. B) Flow diagram illustrating how objects are passed between different module types: 'data frame' - an dataframe of occurrence records; 'raster' - a RasterStack object of the covariates; 'model' - a ZoonModel object, generating standardised predictions from a given model. C) The flow diagram implied by chaining two 'process' modules. D) The flow diagram implied by listing three 'model' modules. Full details of module inputs and outputs, and the effects of listing and chaining each module type are given in the zoon vignette 'Building a module'.

The following code uses `workflow` to run simple a presence-background SDM for a mosquito species in the UK, fitting a MaxEnt model with default settings and 500 randomly placed background points.

```
mosquito1 <- workflow(occurrence = UKAnophelesPlumbeus,
                      covariate  = UKBioclim,
                      process    = Background(n = 500),
                      model      = MaxEnt,
                      output     = InteractiveMap)
```

Chains, Lists and Replicates

Many SDM analyses apply more than one method in one of the five SDM steps, for example to combine multiple covariates and processing steps, to compare various models on the same dataset, or to run the same procedure for several species. zoon enables these more complex workflows by enabling users to pass multiple modules of each type via the `list`, `Chain` and `Replicate` functions.

Chain

`Chain` runs multiple modules of the same type *sequentially*, as illustrated in Figure 1C. For example, in a presence-only analysis a chain of two process modules could be used to generate background data and then to standardise the covariate rasters using the presence/background dataset. Chains can also be applied to occurrence or covariate modules, to combine multiple datasets (e.g. occurrences from different databases) into one. Chaining output modules simply runs each module separately, allowing the user to create multiple maps and summary figures, calculate performance metrics and create other model outputs in one workflow. Model modules are the only module type that may not be chained, since their outputs and inputs are always different.

list

`list` splits a workflow, with each path using a different one of the listed modules, as illustrated in Figure 1D. For example listing multiple model modules would take the same occurrence and covariate data and fit three separate models, applying the same output modules to each model separately. Lists can also be used to run the same SDM procedure for multiple species, or for multiple sets of covariates, or to compare different process modules. Listing output modules causes each module to be applied separately.

Replicate

Some steps in SDM analyses are stochastic and need to be run multiple times; `Replicate` enables this by generating a list with one module repeated a given number of times. `Replicate` could be used to run the same workflow for hundreds of simulated occurrence or covariate datasets, to generate multiple bootstraps for modelling, or to fit models with stochastic elements.

The following workflow uses a chain to standardise the covariate rasters as well as generating background records, and uses a list to fit three different models: MaxEnt, Boosted Regression Trees (also known as Generalized Boosted Regression Models; GBM) and RandomForest. The results of the three models are then returned in three Shiny apps (created by the `Appify` module) allowing the user to interactively explore the occurrence and covariate data, model summaries and prediction maps.

```
mosquito2 <- workflow(occurrence = UKAnophelesPlumbeus,
                      covariate  = UKBioclim,
                      process    = Chain(Background(n = 500),
                                         StandardiseCov),
                      model      = list(MaxEnt,
                                        GBM,
                                        RandomForest),
                      output     = Appify)
```

Cross-validation and external validation

`workflow` handles cross-validation and external validation internally, storing this information in the datframe of occurrence records. *Occurrence* or *process* modules can be used to assign records to one or more cross-validation folds (positive integers indicating different hold-out groups) or to an external validation dataset (indicated by a zero). At the *model* stage, `workflow` fits a separate model for each cross-validation fold then makes and stores predictions for the hold-out datapoints, as well as fitting a model using the whole dataset. In all model fitting, records flagged for external validation are omitted. These cross-validation predictions can then be subsequently analysed by *output* modules to estimate out-of sample predictive performance. Predictions can be also be made from the full model to an external validation dataset and evaluated.

Sharing workflows

The `zoonWorkflow` object returned by `workflow` contains all of the data, code and results used in each stage of the analysis. The object is therefore a self-contained representation of an entire analysis which can be saved as a binary `RData` object, archived, shared with colleagues or uploaded to the web. Anyone else may then load the object into their R session, load the zoon R package, and investigate the analysis.

Workflow objects could therefore be provided as supplementary information to journal articles, or hosted online to meet journal requirements for public archiving of data and software. zoon provides the `ZoonFigshare` function to facilitate sharing a completed workflow object from within R, via the free web platform figshare. `ZoonFigshare` takes a workflow object and some minimal metadata and uploads the workflow as an RData object, along with a metadata text file, to the user's figshare profile for others to download, inspect and modify.

Exploring workflows

`zoonWorkflow` objects are simply R lists containing the outputs from each module in the workflow, as well as the modules used, their versions and arguments and information about the R session in which the workflow was run.

The overall structure of a `zoonWorkflow` object can be inspected and visualised using the provided `print` and `plot` methods. The outputs and intermediate steps of the analysis can be extracted directly from the `zoonWorkflow` object, or by using the utility functions `Occurrence`, `Covariate`, `Process`, `Model` and `Output` to extract the outputs of each analytical stage. These functions return either the single R object outputted by each module, or list of objects when a list of modules was used in the workflow.

Reproducing and extending workflows

Given a `zoonWorkflow` object, the analysis can be repeated in its entirety using the zoon function `RerunWorkflow`. This function can therefore be used to rapidly update an analysis whenever the underlying occurrence or covariate data (from an online repository for example) are updated. Used in this way the function is equivalent to copying the authors original source code. `RerunWorkflow` can also run from a specific stage of the workflow, for example using the previously downloaded data stored in the object but re-running an output module to recreate a plot.

Workflows can similarly be modified and re-run by replacing or adding modules in one or more of the five

analytical steps using the function `ChangeWorkflow`. `ChangeWorkflow` only re-runs the changed modules and those downstream; the stored outputs of earlier modules are re-used rather than re-running them.

This functionality therefore enables researchers to explore and alter existing workflows, such as from a published analysis, without having to re-run computationally expensive models or re-download datasets that may have changed in the interim. See Example 2 below for a demonstration of this functionality.

Exploring available modules

zoon modules are not distributed with the zoon R package, but are instead downloaded on-the-fly from the online module repository. New modules uploaded to the repository therefore become instantly available to zoon users, without having to update the R package. zoon can query the currently available modules of each type via the function `GetModuleList`.

All zoon modules on the repository are accompanied by documentation and metadata, similarly to R's helpfiles. The documentation for any module on the repository can be accessed from R using the function `ModuleHelp`. zoon also provides a function `ZoonCitation` to provide citation information, similarly to the `citation` function provided with R for citing R packages.

The only pre-requisite for uploading a module to the online repository is that it passes a series of automated unit and integration tests to ensure interoperability with other modules. Similarly to the CRAN archive of R packages, zoon leaves the SDM community to determine which methods and software should be used, rather than acting as a gatekeeper of SDM methods.

A web service for interactive exploration of available modules and their documentation is currently being developed. This platform may include systems by which users can rate different modules, making it easier for the community to promote the best modules, flag methodological issues, and collaborate on module development.

Example Applications

Next we demonstrate how zoon can be used in practice for running SDMs, creating and sharing new modules, and performing a methods comparison. The workflow objects created by these analyses can be accessed at `figshare.com/articles/zoon_applications_paper_workflows/4597792`. We encourage readers to download, interrogate and alter these workflows for themselves.

Example 1. Modelling the potential distribution of nine-banded armadillo

Feng & Papeş (2015) constructed a MaxEnt species distribution model for nine-banded armadillo using presence-only data on the species' current distribution, and the bioclim (Hijmans *et al.* 2005) set of environmental correlates. This model was then used to predict areas in the Americas which may be suitable for the species to become established.

Such a model can be quickly and easily re-constructed as a zoon workflow using modules available in the zoon module repository. Feng & Papeş (2015) used a combination of occurrence data from GBIF, and additional occurrence data manually collected from the published literature. Unfortunately the latter data have not been made publically available, so here we use only data from GBIF. If the additional data had been made available it would be straightforward to incorporate them, for example using the `LocalOccurrenceData` module.

```
FengPapes <-
  workflow(occurrence = SpOcc('Dasypus novemcinctus',
                              extent = c(-130, -20, -60, 60)),
           covariate = Bioclim(extent = c(-130, -20, -60, 60),
                               layers = c(1:4, 6, 9, 10, 12, 15)),
           process = Chain(Clean,
                           MESSMask,
                           Background(n = 10000,
                                      bias = 200),
                           Crossvalidate(k = 5)),
           model = MaxEnt,
           output = PrintMap(points = FALSE,
                             threshold = 0.05,
                             thresholdmethod = 'falsenegative',
                             xlim = c(-130, -70),
                             ylim = c(20, 50)))
```

This workflow plots a static map of the predicted distribution. This map is shown in Figure 2A and corresponds to Figure 3 in Feng & Papeş (2015). The resulting workflow contains all the code required to re-run the workflow, the input data and the results of executing each module. The object `FengPapes` could therefore be saved as a binary file and shared as a reproducible representation of this research.

Next, we update the workflow to produce an interactive map enabling anyone to inspect the data and predictions on a zoomable map, and to inspect the response curves of the fitted model. These outputs are shown in Figure 2, panels B and C. Various other output modules could be used here, e.g. to validate models, project distributions to new regions, or save prediction maps in a variety of formats.

```
FengPapesUpdate <-
  ChangeWorkflow(workflow = FengPapes,
                 output = Chain(ResponseCurve(cov = 1),
                                InteractiveMap))
```

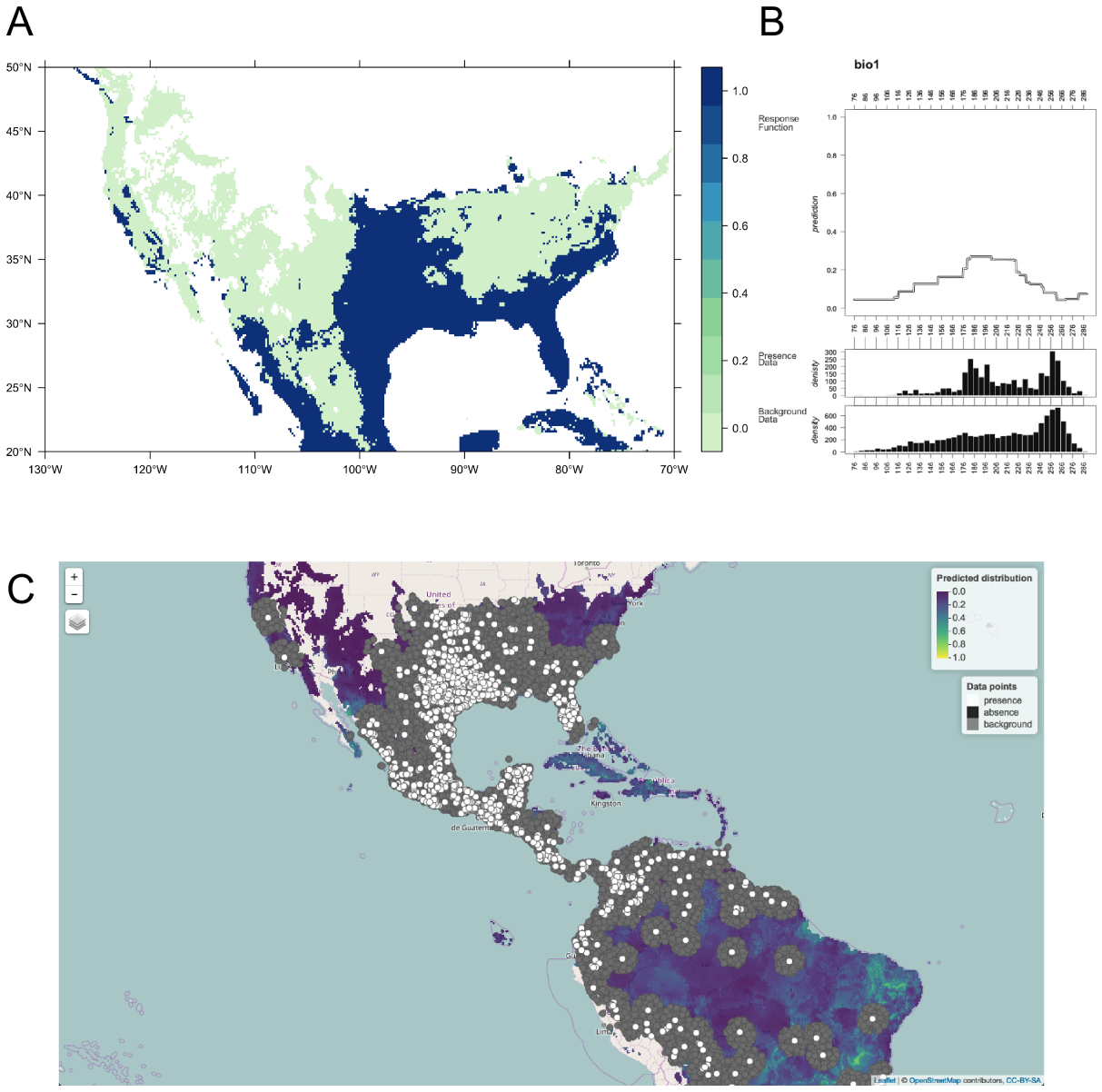

Figure 2: Outputs of the workflow objects 'FengPapes' and 'FengPapesUpdate'. A) Map of the MaxEnt predicted distribution, with a 5% omission rate threshold, by the 'PrintMap' module in the workflow 'FengPapes' which encodes the core of a published analysis. B) A response curve produced by the 'ResponseCurve' module for the first covariate, bio1 in the workflow 'FengPapesUpdate', which modifies the original analysis workflow. C) A screenshot of the interactive map produced by the 'InteractiveMap' modules in the workflow 'FengPapesUpdate', displaying raw occurrence data and predicted distribution over a global map, allowing users to interactively explore their results. White areas are masked due to being in the MESS mask. Any SDM analysis distributed as a zoon workflow can be easily be explored and scrutinized by modifying its output modules using the function 'ChangeWorkflow'.

Example 2. Building a spatial thinning module

Aiello-Lammens *et al.* (2015) proposed an approach for dealing with spatial sampling bias in presence-only data by 'thinning' the presence data, and provide an R package `spThin` to implement their procedure (Aiello-Lammens *et al.* 2014). We can incorporate this approach in a workflow, by defining a simple *process* module that adapts the zoon data into the spThin format, uses this package to apply the algorithm, and converts data back into zoon's expected format again:

```
spThin <- function (.data, thin = 50) {

  # check these are presence-background data
  stopifnot(all(.data$df$type %in% c('presence', 'background')))

  # install & load the package
  zoon::GetPackage('spThin')

  # get dataframe & index to presence data
  df <- na.omit(.data$df)
  pres_idx <- which(df$type == 'presence')

  # prepare presence data subset and apply thinning
  sub_df <- data.frame(LAT = df$latitude[pres_idx],
                       LONG = df$longitude[pres_idx],
                       SPEC = NA)
  th <- thin(loc.data = sub_df,
             thin.par = thin,
             reps = 1,
             locs.thinned.list.return = TRUE,
             write.files = FALSE,
             write.log.file = FALSE)

  # get index to rows in sub_df, update the full dataset and return
  pres_keep_idx <- as.numeric(rownames(th[[1]]))
  .data$df <- rbind(df[pres_idx,][pres_keep_idx, ],
```

```
                    df[-pres_idx, ])
  return (.data)
}
```

To convert this code into a zoon module, we need to write it to a standalone file (named `spThin.R`) with the metadata required to build the module documentation. The zoon function `BuildModule` helps with this step, and can also run checks to make sure we got everything right:

```
BuildModule(object = spThin,
            type = 'process',
            title = 'Spatial thinning of Presence-only Data',
            description = paste('Apply the stochastic spatial thinning',
                                'algorithm implemented in the spThin',
                                'package to presence data in a',
                                'presence-background dataset'),
            details = paste('Full details of the algorithm are available in',
                            'the open-access article by Aiello-Lammens',
                            'et al. (2015): dx.doi.org/10.1111/ecog.01132'),
            author = 'zoon Developers',
            email = 'zoonproject@gmail.com',
            paras = list(thin = paste('Thinning parameter - the required',
                         'minimum distance (in kilometres) between points',
                         'after applying the thinning procedure')),
            dataType = 'presence-only',
            check = TRUE)
```

This module can now be shared so that others can use it in their zoon workflows. Modules can be uploaded to the zoon modules repository via the online submission system at [URL TBD]. The zoon package vignette *Building modules* provides full details and examples of how to build modules of each type.

Example 3. Unpacking MaxEnt

The popular MaxEnt SDM model has recently been shown to be equivalent to a Poisson point process model (Renner & Warton 2013), and to be closely approximated by a logistic regression model with weights applied to background datapoints (Warton & Shepherd 2010; Fithian & Hastie 2013; Renner *et al.* 2015). Given

this close correspondence between MaxEnt and logistic regression, Renner & Warton (2013) and others have suggested that MaxEnt's superior predictive performance (Elith *et al.* 2006) is most likely due to the array of features (candidate covariate transformations) it constructs and its use of regularisation to prevent overfitting.

zoon enables us to investigate this hypothesis by easily comparing MaxEnt models fitted with and without regularisation and feature construction. We fit MaxEnt with both the widely used java implementation (omitting threshold features) as well as the downweighted logistic regression approximation provided in the `maxnet` R package (Phillips 2016). We also compare these models with a standard logistic regression model which does not apply the downweighting step. The following workflow fits these models to a previously published presence-background data on the Carolina wren in the USA (Royle *et al.* 2012), maps their predictions (Figure 3) and evaluates the predictive performance of each model against the full presence-absence dataset by AUC. Note that setting the regularisation constant in the `maxnet` package to zero caused numerical errors, so we use a small value instead.

```
MaxEntComparison <- workflow(
  occurrence = CarolinaWrenPO,
  covariate  = CarolinaWrenRasters,
  process    = Chain(SubsampleOccurrence(500),
                     Background(n = 10000),
                     CarolinaWrenValidation),
  model      = list(MaxEnt(args = 'threshold=false'),
                    MaxNet,
                    MaxNet(regmult = 0.005),
                    MaxNet(features = 'l'),
                    MaxNet(regmult = 0.005, features = 'l'),
                    LogisticRegression),
  output     = Chain(PrintMap(points = FALSE),
                     PerformanceMeasures))
```

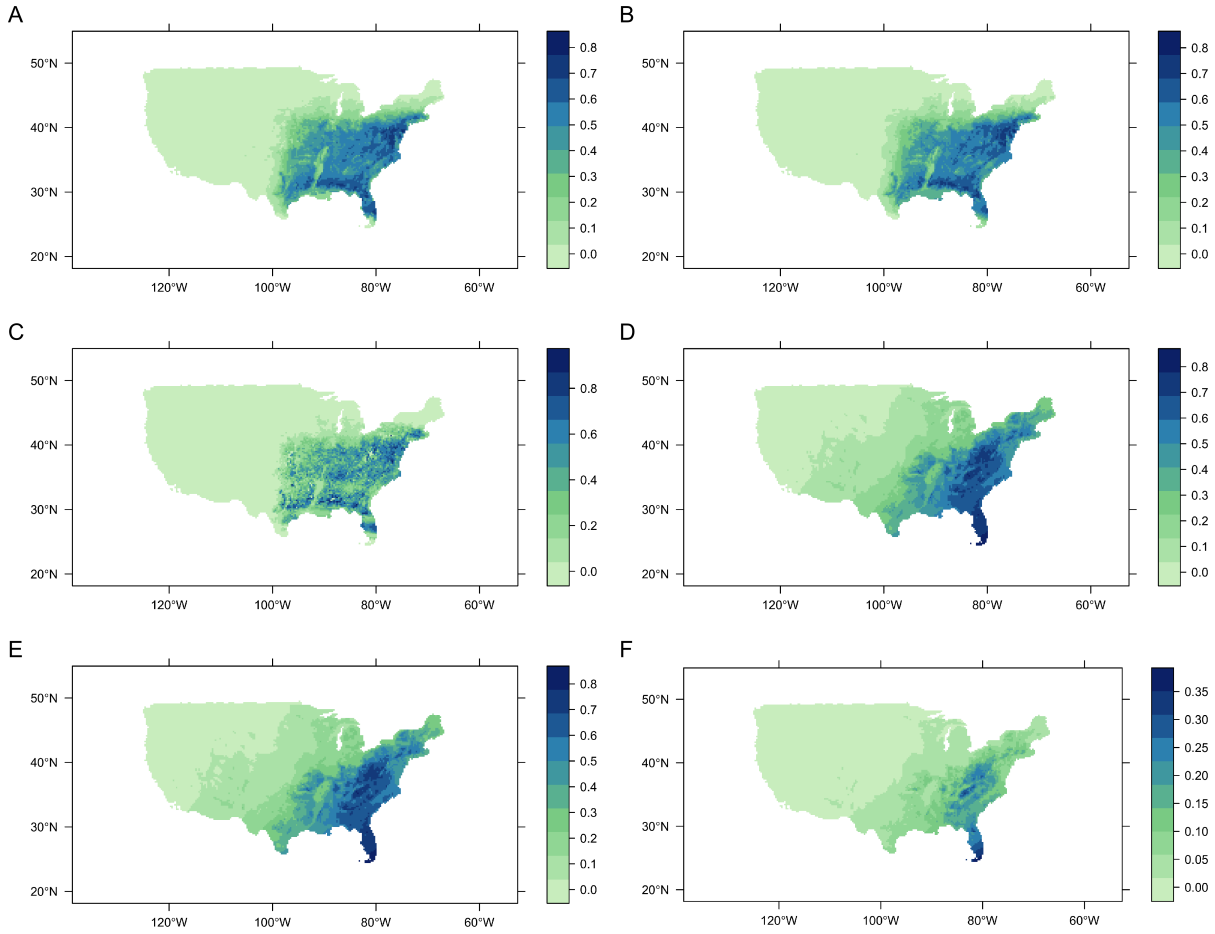

Figure 3: Prediction of the distribution of the Carolina wren from 6 different models, produced by the workflow 'MaxEntComparison'. A) MaxEnt without threshold features. B) MaxNet with default settings. C) MaxNet without regularisation. D) MaxNet with regularisation but only linear features. E) MaxNet without regularisation and only linear features. F) Logistic regression. MaxNet with full features but no regularisation (C) gave the most local complexity, indicative of overfitting to the data. The models with only linear features (D-F) had a broader distribution, indicative of underfitting. Differences between MaxNet with linear features and no regularisation (E) and logistic regression (F) are due to the downweighting applied to background data in the former.

The AUC statistics calculated against the presence-absence data were: MaxEnt 0.9622; MaxNet 0.966; MaxNet (no regularization) 0.9568; MaxNet (linear features) 0.9372; MaxNet (linear features and no regularization) 0.9375; Logistic regression 0.9389. As expected, MaxEnt and MaxNet (the logistic regression approximation) generated similar predictions and had similar predictive performance. Likewise, once the regularisation and features of MaxNet were switched off, both the predictions and performance were very similar to logistic regression.

Readers interested in a more comprehensive comparison, or in exploring the response curves, can easily reproduce and modify the analysis either by editing and re-running the above code block or by downloading

the workflow object and using the `ChangeWorkflow` function.

Future developments

Reproducibility and open science are gaining significant attention (Borregaard & Hart 2016), and journals, funders and governments are increasingly making them a requirement (Obama 2013; European Commission 2016; McNutt 2016). Researchers must therefore find ways of at least disseminating code and data, if not making them usable. Despite numerous platforms for archiving and sharing code and data, users still face a challenge locating and using resources across different platforms. Increasing the usability of code and data is therefore the real challenge of open and reproducible science.

The size and fractured nature of the SDM research community therefore warrants a discipline-specific open research platform such as zoon to resolve these problems. By providing a common interface for SDM and the capacity to create and share methodological experiments zoon offers a new opportunity for the SDM community to develop community modelling benchmarks, and resolve scientific and methodological questions in ways that the current culture of publishing cannot achieve.

zoon also has the potential to improve methodological standards in SDM, a research area in which there is widespread misunderstanding and misuse of techniques (Yackulic *et al.* 2013). SDM methodologists could encode model validation procedures and best-practice guidance as zoon output modules, enabling end users to rapidly ensure their modelling procedures are fit for purpose (Guillera-Arroita *et al.* 2015).

zoon was both conceived and developed based on input from users. Continuing and expanding user engagement will be vital to forming a critical mass of ecologists creating and using zoon modules in their research. To assist users a number of tutorials on using zoon and developing modules are provided on the zoonproject GitHub repository and as vignettes distributed with zoon. We intend to supplement these with a gallery of research examples and a forum where the community can discuss and evaluate best SDM practice in a reproducible way.

Supporting Information

Version 0.6 of the zoon R package is archived at https://doi.org/10.5281/zenodo.240926. The code required to reproduce this manuscript can be downloaded at https://github.com/zoonproject/zoon_app_paper (to be replaced with zenodo archive link after proofing stage).

Acknowledgements

The 2020 Science programme is funded through the EPSRC Cross-Disciplinary Interface Programme (grant number EP/I017909/1).

References

Ahmed, S.E., Mcinerny, G., O'Hara, K., Harper, R., Salido, L., Emmott, S. & Joppa, L.N. (2015). Scientists and software - surveying the species distribution modelling community. *Diversity and Distributions*, **21**, 258–267.

Aiello-Lammens, M.E., Boria, R.A., Radosavljevic, A., Vilela, B. & Anderson, R.P. (2015). spThin: An R package for spatial thinning of species occurrence records for use in ecological niche models. *Ecography*, **38**, 541–545.

Aiello-Lammens, M.E., Boria, R.A., Radosavljevic, A., Vilela, B. & Anderson, R.P. (2014). *SpThin: Functions for spatial thinning of species occurrence records for use in ecological models*. Retrieved from https://CRAN.R-project.org/package=spThin

Araújo, M.B. & Peterson, A.T. (2012). Uses and misuses of bioclimatic envelope modeling. *Ecology*, **93**, 1527–1539.

Araújo, M.B., Thuiller, W. & Yoccoz, N.G. (2009). Reopening the climate envelope reveals macroscale associations with climate in European birds. *Proceedings of the National Academy of Sciences*, **106**, E45–E46.

Barbosa, F.G. & Schneck, F. (2015). Characteristics of the top-cited papers in species distribution predictive models. *Ecological Modelling*, **313**, 77–83.

Beale, C.M., Lennon, J.J. & Gimona, A. (2009). European bird distributions still show few climate associations. *Proceedings of the National Academy of Sciences*, **106**, E41–E43.

Beale, C.M., Lennon, J.J. & Gimona, A. (2008). Opening the climate envelope reveals no macroscale associations with climate in European birds. *Proceedings of the National Academy of Sciences of the United States of America*, **105**, 14908–14912.

Borregaard, M.K. & Hart, E.M. (2016). Towards a more reproducible ecology. 349–353.

Boulton, G., Campbell, P., Collins, B., Elias, P., Hall, W., Laurie, G., O'Neill, O., Rawlins, M., Thornton, J.,

Vallance, P. & others. (2012). Science as an open enterprise. *The Royal Society*.

Cooper, J., Scharm, M. & Mirams, G.R. (2016). The cardiac electrophysiology web lab. *Biophysical journal*, **110**, 292–300.

De Giovanni, R., Williams, A.R., Ernst, V.H., Kulawik, R., Fernandez, F.Q. & Hardisty, A.R. (2016). ENM Components: A new set of web service-based workflow components for ecological niche modelling. *Ecography*, **39**, 376–383.

De Giovanni, R., Williams, A.R., Ernst, V.H., Kulawik, R., Fernandez, F.Q. & Hardisty, A.R. (2015). ENM Components: a new set of web service-based workflow components for ecological niche modelling. *Ecography*, 1–8.

Elith, J., Graham, C.H., Anderson, R.P., Dudik, M., Ferrier, S., Guisan, a., Hijmans, R.J., Huettmann, F., Leathwick, J.R., Lehmann, a., Li, J., Lohmann, L.G., Loiselle, B.a., Manion, G., Moritz, C., Nakamura, M., Nakazawa, Y., McC., O.J., Peterson, a.T., Phillips, S.J., Richardson, K.S., Scachetti-Pereira, R., Schapire, R.E., Soberon, J., Williams, S., Wisz, M.S. & Zimmermann, N.E. (2006). Novel methods improve prediction of species' distributions from occurrence data. *Ecography*, **29**, 129–151.

European Commission. (2016). Open access to scientific publications and research data in horizon 2020.

Feng, X. & Papeş, M. (2015). Ecological niche modelling confirms potential north-east range expansion of the nine-banded armadillo (Dasypus novemcinctus) in the USA. **42**, 803–807.

Fithian, W. & Hastie, T. (2013). Finite-sample equivalence in statistical models for presence-only data. *Annals of Applied Statistics*, **7**, 1917–1939.

Guillera-Arroita, G., Lahoz-Monfort, J.J., Elith, J., Gordon, A., Kujala, H., Lentini, P.E., Mccarthy, M.A., Tingley, R. & Wintle, B.A. (2015). Is my species distribution model fit for purpose? Matching data and models to applications. *Global Ecology and Biogeography*, **24**, 276–292.

Hijmans, R.J., Cameron, S.E., Parra, J.L., Jones, P.G. & Jarvis, A. (2005). Very high resolution interpolated climate surfaces for global land areas. *International journal of climatology*, **25**, 1965–1978.

Hijmans, R.J., Phillips, S., Leathwick, J. & Elith, J. (2016). *Dismo: Species distribution modeling*. Retrieved from https://CRAN.R-project.org/package=dismo

Joppa, L.N., McInerny, G., Harper, R., Salido, L., Takeda, K., O'Hara, K., Gavaghan, D. & Emmott, S.

(2014). Troubling Trends in Scientific Software Use. *Science*, **340**, 814–815.

McNutt, M. (2016). Taking up top. *Science*, **352**, 1147–1147.

Naimi, B. & Araújo, M.B. (2016). sdm: a reproducible and extensible R platform for species distribution modelling. *Ecography*, **39**, 1–8.

Obama, B. (2013). Executive order–Making open and machine readable the new default for government information. *The White House*.

Peng, R. (2011). Reproducible Research in Computational Science. *Science*, **334**, 1226.

Phillips, S. (2016). *Maxnet: Fitting 'maxent' species distribution models with 'glmnet'.*

Phillips, S.J., Anderson, R.P. & Schapire, R.E. (2006). Maximum entropy modeling of species geographic distributions. *Ecological modelling*, **190**, 231–259.

R Core Team. (2016). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. Retrieved from https://www.R-project.org/

Renner, I.W. & Warton, D.I. (2013). Equivalence of MAXENT and Poisson Point Process Models for Species Distribution Modeling in Ecology. *Biometrics*, **69**, 274–281.

Renner, I.W., Elith, J., Baddeley, A., Fithian, W., Hastie, T., Phillips, S.J., Popovic, G. & Warton, D.I. (2015). Point process models for presence-only analysis. *Methods in Ecology and Evolution*, **6**, 366–379.

Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F. & Dormann, C.F. (2016). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography*, n/a–n/a. Retrieved from http://dx.doi.org/10.1111/ecog.02881

Royle, J.A., Chandler, R.B., Yackulic, C. & Nichols, J.D. (2012). Likelihood analysis of species occurrence probability from presence-only data for modelling species distributions. *Methods in Ecology and Evolution*.

Thuiller, W., Lafourcade, B., Engler, R. & Araújo, M.B. (2009). BIOMOD–a platform for ensemble forecasting of species distributions. *Ecography*, **32**, 369–373.

Warton, D.I. & Shepherd, L.C. (2010). Poisson point process models solve the 'pseudo-absence problem' for presence-only data in ecology. *Annals of Applied Statistics*, **4**, 1383–1402.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A.,

441 Balcazar Vargas, M.P., Sufi, S. & Goble, C. (2013). The Taverna workflow suite: designing and executing
442 workflows of Web Services on the desktop, web or in the cloud. *Nucleic acids research*, **41**, 1–5.

443 Yackulic, C.B., Chandler, R., Zipkin, E.F., Royle, J.A., Nichols, J.D., Campbell Grant, E.H. & Veran, S.
444 (2013). Presence-only modelling using MAXENT: When can we trust the inferences? *Methods in Ecology and*
445 *Evolution*, **4**, 236–243.